Super-Utility Disk V2.0



Moltkestr. 29
5620 Velbert 1
Tel.:02051/84815
oder 84410

1.0Macroass

1.1 Einführung

Macroass ist ein kompletter 2-Pass Macro-Assembler inclusive Editor. Geschrieben wurde er zum großen Teil in Basic und ca. 1K in Assembler (mit einer Vorgängerversion). Es soll daher hier gleich betont werden das "Macroass" zwar vieles kann, was auch seine "großen Brüder" können die rein in Maschienensprache geschrieben sind, aber der größte Nachteil die langsamere Assembliergeschwindigkeit ist.
Dafür kann er aber schon Bildschirmcode als Parameter erzeugen (z.B. LDA #!A) . Desweiteren, ist er Macro- fähig, wobei sogar eine, wenn auch eingeschränkte Parameterübergabe an den Macrocode möglich ist. Da versucht wurde Macroass möglichst kompremiert zu programmieren, hat man noch genug Platz für über 500 Zeilen Quellcode und 1K vorassemblierten Macrocode, so daß man mit Macroass durchaus kleine bis mittel- lange Maschienenprogramme schreiben kann.
Zugleich muß hier noch gesagt werden, daß die Speicheroptimierte Programmierweise auch Nachteile mit sich bringt. Da beim Assemblieren keine Speicheraufwendigen Tabellen angelegt weredn (z.B. für Labels), muß im Bedarfsfall immer wieder neu nach einem Label gesucht werden, was natürlich auf Kosten der Geschwindigkeit geht.
Da nicht jeder erdenkliche Fehler abgefangen wird, hat es ein geschickter Spaßvogel auch nicht schwer den Assambler hereinzulegen. Solange es sich hierbei nicht um einen ganz harten Fall handelt, versucht Macroass immer das beste aus dem Quellcode zu machen. Ausserdem ist Macroass aus Speichergründen nicht in der Lage die Parameter bei Macros so elegant zu übergehen, wie manch ein größeres Vorbild.

1.2 Zur Beachtung

Mit dem Macroass Editor, kann man fast so umgehen wie man es vom Basic-Editor her kennt. Was zu beachten wäre ist,

1. Zeilennummern müssen immer innerhalb der ersten 4 Positionen einer Zeile stehen.
2. Ein Label muß bei Position 6 anfangen.
3. Zwischen Label und Mnemonic und zwischen Befehl und Parameter muß mindestens ein Leerzeichen stehen.
4. Der Tabulator muß passend auf die verbindliche Labelposition und die vorgeschlagene Mnemonic eingestellt sein.

1.3 Befehle des Editors

Der Editor versteht neben dem Editieren von Zeilen noch 17 Kommandos, die hier nun wie folgt erklärt werden.

1. L oder Lxxx: Listet den eingetippten oder geladenen Quellcode geordnet nach der Zeilennummer . Gegebenenfalls ab Zeile xxx
2. Rxxx,yyy: Renummeriert den gesamten Quellcode. Startnummer xxx, Abstand yyy.
3. FRE: Gibt die Anzahl der maximal noch einzugebenden Zeilen an.
4. NEW: Löscht den Quellcode.
5. NUM: Aktiviert/Deaktiviert die Autonumber-Funktion Nach Eingabe einer Zeile wird eine um 10 höhere Nummer ausgegeben. (Auch zum löschen größerer Blöcke geeignet. Startnummer eingeben und Return).
6. DIR: Gibt das Direktory von D1: *.* aus.
7. PRINT: Gibt den Quellcode auf dem Drucker aus, falls angeschlossen. (Wird beim Laden überprüft)
8. LOAD: Speichert den Ouellcode.
9. SAVE: Lädt den Queollcode.
10. ASM: Ruft den Assembler auf (Achtung lange Wartezeit).
11. § : Gibt den erzeugten Code auf dem Bildschirm aus.

12. MEM: Poket den erzeugten Code bei einer anzugebenden Adressse in den Speichrer.
13. SAVEC: Savet den erzeugten Code.
14. MLOAD: Lädt vorassemblierte Macros.
15. MSAVE: Savet die im Macrobuffer befindlichen Macros.
16. CLRM: Löscht den Macrobuffer.
17. ML: Listet alle im Buffer befindlichen Macros inclusive ihrer Codelänge auf.

1.4 Der 2-Pass-Assembler

Der 2-Pass Assembler wird mit ASM aufgerufen. Im ersten Pass werden hauptsächlich die absoluten Adressen zu den jeweiligen Quellcoden berechnet. Im zweiten Pass wird dann übersetzt. Bei den Mnemonics versteht der Macroass die übliche Standardsyntax.

1.4.1 Die Parameter und ihre Operatoren

$: gefolgt von einer 2- oder 4-stelligen Hexzahl
%: gefolgt von einer 8-stelligen Binärzahl
´: berechnet den ASCII-Wert eines Zeichens
!: berechnet den Bildschirmcode für ein Zeichen.
>: selektiert das HIGH-Byte
<: selektiert das LOW-Byte

1.5 Macros

Unter Macro versteht man im Zusammenhang mit Assembler ein meist kurzes Assemblerprogramm, bei dem der Macrodeffinition ein Name zugewiesen wird. Man kann Macros auch auch als Userdefinierte Assembler-befehle bezeichnen. Immer wenn im Assemblerprogramm (Quellcode) dieser Name bzw. Befehl auftaucht wird an dieser Stelle die besagte kurze Assemblersequenz eingesetzt. Auf diese Weise bekommt man hinterher ein sehr übersichtliches und gut lesbares Assemblerprogramm (besonders wichtig bei der Fehlersuche). Es ist daher praktisch, einen großen Teil von z.B. BASIC-Kommandos in Form von Macros vorzudefinieren.
Macronamen können zwar beliebig lang sein, jedoch beachtet der Macroass nur die ersten 3 Zeichen nach dem *: Alle Macros die vom Assembler als solche erkannt werden, werden in den Macrobuffer eingetragen, der auch ein NEW überdauert. Er wird erst mit CLRM vollständig gelöscht. Mit ML kann man den Inhalt des Macrobuffers abfragen.
Ein Programm das gerade assembliert wird, hat also nicht nur Zugriff auf etwaige im eigenen Quellcode befindliche Macros, sondern auf alle im Buffer befindlichen Macros. Man kann sich deshalb ein File erstellen, welches nur Macros enthält, dieses assemblieren und sowohl Quellcode wie auch die fertig assemblierten Macros mit MSAVE sichern. Bei Bedarf kann man nun die fertigen Macros mit MLOAD einladen. Das verkürzt die Assemblierzeit und spart Platz für den Quellcode.
Zusammenfassend hat man also die Möglichkeit, sich eine komplette Befehlserweiterung zu schreiben! Diese müssen dann bei Abruf nicht mehr neu assembliert werden.

1.6 DATA

DATA : Der Macroass unterscheidet im Gegensatz zu vielen anderen Assemblern nicht zwischen WORD und BYTE, sondern er verfügt über ein allgemeines DATA Kommando.
DATA macht aus seinen Parametern grundsätzlich Bytes, es sei denn das Parameter wurde mit dem ^°-Zeichen ausdrücklich als 2 Byte WORD deklariert.

1.6.1 Parameterübergabe

Die Parameterübergabe erfolgt bei Macroass als 2Byte Worte, über die als Parameterübergaberegister festgeleg- tenSpeicherzellen 214-219. Über die festdefinierten LABELs PARAM1 bis PARAM3 bzw. PARAMX kann man dann leicht darauf zugreifen. Die Parameter werden also der Reihe nach in besagten Adressen abgelegt.

1.7 Beispiele

1.7.1 Definition von Macros

```
1000      *MACRO NAME
1010      Assemblerprogramm
1020      *ENDM
```

1.7.2 Macro für Pause-Kommando

```
1000 ;
1010      *MACRO PAUSE
1020      LDY PARAM1
1030 XLP  LDX #255
1040 DX   DEX
1050      BNE DX
1060      DEY
1070      BNE XLP
1080      *ENDM
1090;
```

1.7.3 Aufrufen des Pause-Macros

```
1310 ;
1320 TON  LDA #8
1330      STA 53279
1340      *PAUSE 10    ;MACRO AUFRUF
1350      LDA #0
1360      STA 53279
1370      *PAUSE 10    ;MACRO AUFRUF
1380      JMP TON
1390;
```

1.7.4 Allgemeine Aufrufsyntax

```
1000;
1010      *MACRONAME
1020;
```

1.7.5 Aufrufsyntax mit Parametern

```
1000 ;
1010      *MACRONAME DATA1,DATA2,..........
1020 ;
```

1.7.6 DATA

```
1000;
1010      DATA 255,255,^°STARTADR,^°END-1
1020;
1030      DATA  A, ^° $1234,%11111111,^°2
1040;
```

1.7.7 Startadresse setzen

```
1000;
1010 *= $0600
1020;
```

1.8 Weitere Pseudo-Assemblerbefehle mit Beispielen

EQU : Entspricht dem -Zeichen.
```
1000;
1010ANFANG EQU 100
1020;
```

ASC : Baut ASCII-Strings in den Code ein.
```
1000;
1010        ASC "Bitte Disk einlegen"
1020;
```

COD : Baut Bildschirmcode in den Code ein.
```
1000;
1010        COD "PLAYER 1"
1020;
```

2.0 DL-Designer

2.1 Einführung

Der DL-Designer ist ein Programm zur ist ein Programm zur Erstellung von anwenderspezifischen Display-Lists. Das heißt, es können Bildschirme aufgebaut werden die aus (fast) beliebig vielen Kombinationen der Grafikmodi bestehen. Die Mischuntg der Modi erfolgt dabei zeilenweise.
Die Ausgabe der fertigen DL erfolgt entweder als Zahlen, ASCII-Zeichen oder als fertiges Basic Programm. Letztere Möglichkeit ist vor allen Dingen für " Anfänger " geeignet. Natürlich können Sie Ihre fertigen DLs dann auch speichern und zu gegebenem Zeitpunkt neu aufrufen.

2.2 Programmbedienung

Nach dem Titel, den man mit " START " verlassen kann, hat man zwei Möglichkeiten. Entweder man aktiviert die Automatik, bei der man sich um die berechnung von Adressen unsw. nicht zu kümmern braucht, und nur die gewünscte Grafikmodi der Reihe nach eingeben muß. Oder man berechnet die bei LMS und JVB benötigten Adressen " einfach " selbst.
In der Bildschirmmitte, sehen Sie nun alle Grafikstufen aufgezählt.
" L " steht für Grafiklose Leerzeilen wobei pro Aufruf 1 - 8 möglich sind.
" GTM " steht für GTIA Modi, das heißt GR.9 - 11.Von diesen kann pro Aufruf je nur eine gewählt werden. Außerdem kann nur entweder nur ein GTIA-Modus oder GR.8 gewählt werden. Beide zusammen ist nicht ohne weiteres möglich.

2.2.1 Anzeigen

Rechts neben dem Selektionsteil sind die Anzeigen " SC-MEM " (Screen-Memory), in der der bisher benötigte Bildspeicher angezeigt wird. Und die Anzeige " SLC " (Screen-Line-Counter) in dieser werden die benutzten Scan-Lines gezählt. Auf dem linken Anzeigebalken können Sie erkennen wie weit Sie ihren Bildschim schon definiert haben.

2.2.2 Menü

Am unteren Rand shen Sie noch ein Menuemit diversen Funktionen. Um diese Funktionen brauchen Sie sich nicht zu kümmern, wenn Sie nicht gerade " Profi " sind. Es würde hier aus dem Rahmen fallen, die gesamte Befehlsstruktur einer DL zu erklären. Lediglich die beiden letzten Menüpunkte sind von Interresse. Wenn die

DL fertig erstellt ist drücken Sie " J " im Falle Sie wollen neu beginnen " N " . Die Taste " SPACE " wird zum Schutz gegen zufälliges Löschen verwand

1.3 Horizontalscroll-LMS Automatik

Ist die DL fertig erstellt, und ist im Automatik Modus können Sie nun bei Bedarf die Horizontal-LMS Automatik zuschalten. Diese berechnet die nötigen LMS-Adressen,je nach gewünschter Zeilenlänge für ein byteweises (grobes) Scrolling. Der " Laie ", ignoriert dieses einfach und verneint " N ". Auf jeden Fall, erfolgt nun eine mehr oder weniger lange Wartezeit, die je nach DL von wenigen Sekunden bis zu einigen Minuten dauern kann. In dieser Zeit berechnet Ihr Computer die LMS und JVB Adressen. Danach, erfolgt auf Wunsch ein Testlauf der DL. Wenn Sie mit dem Ergebnis zufrieden sind, betätigt man " SPACE ". Ist das Ergebnis unzureichend Betätigen Sie " START " und beginnen erneut.

2.4 Ausgabe der DL

Weiter geht es nun mit der Ausgabe der DL. Unter der Funktion Automatik, können Sie die DL als fertiges Basic-Programm ausgeben das Sie dann in ihre eigenen Programme übernehmen können.
Andere Möglichkeiten der Ausgabe sind Zahlen und ASCII-Zeichen. Besonders bei längeren DLs ist es ratsam, diese zu sichern. Der danach ausgegebene Vorschlag zum Wiedereinladen, kann in das oben erwähnte Basic-Programm an statt der Stringdefinition eingesetzt werden.
Hinweis:
Sollte in der Stringdefinition des Basic-Programms mal ein Anführungszeichen auftauchen, was bei der Eingabe einen Syntax-Error verursachen würde, dann zählt man die Position im String ab und ersetzt dieses durch ein anderes Zeichen. Später schreibt man dann: DL$(X,X)=CHR$(34)

3.0 Minidiskmon II

3.1 Einleitung

Minidiskmon II ist ein DOS 2.5 kompatibler Diskettenmonitor,. mit dem einzelne Sektoren direkt editiert werden können. Desweiteren verfügt er über fiele nützliche Funktione wie z.B. FIND und COPY. Gestartet wird der DiskmonII mit RUN "D:DISKMON.BAS"

3.2 Menüfunktionen

M= MENUE : zurück zum Hauptmenü
D= DIREKTORY: lädt die DOS 2.5 Direktory
L= LOAD: lädt einen Sektor und gibt diesen als HEX/ATASCII-DUMP auf dem Bildschirm aus
N= NEXT: lädt den nächsten Sektor eines Files
  PREVIOUS: lädt den Sektor mit der nächstkleineren Nummer
  NEXT: lädt den Sektor mit der nächsthöheren Nummer
S= SAVE: schreibt den gerade als DUMP sichtbaren Sektor auf Diskette (mit Sicherheitsabfrage)
C= COPY: bis zu 100 Sektoren können auf eine andere Diskette oder Position kopiert werden
F= FIND: sucht eine einzugebende Bytesequenz innerhalb eines bestimmten Bereichs
R= RESTORE: der alte Inhalt des zuletzt gesavten Sektors wird aus dem Restorebuffer geladen (FIND zerstört den Restorebuffer !!!)
E= EDIT: der geladene Sektor kann in 5 " Zahlensystemen " editiert werden (CTRL-TAB und Taste D/H/B/C/A drücken)

4.0 QOS

4.1 Einführung

QOS bedeutet " Q " Operating System. Es ist ein Ram-Disk Programm, welches den Speicher unter dem OS-Rom verwaltet.

4.2 Handhabung

Da sich QOS nach dem Booten nicht sebstständig initialisiert was bei einem softwaremäßig ausgelöstem Kaltstart wie z.B. einem Systemabsturz von Vorteil ist, da sich die Daten nach dem Booten von QOS noch erreichen lassen, können Sie das mit folgender Sequenz tun:

```
OPEN# x,y,z,"Q:+":CLOSE#x
```

oder ganz einfach mit:

```
SAVE "Q:+"
```

In den 88 freien Pages ab $A000 von denen eine für Verwaltung abgeht, können mit QOS bis zu 16 Files verwaltet werden. Jeder Filename muß dabei 4 Zeichen lang sein:

```
SAVE "Q:FILE"    LOAD "Q:DATA"    OPEN #1,4,0,"Q:TXT1"
LIST "Q:XY  "    ENTER "Q:  10"   SAVE "Q:T 1 "
```

Wenn man ewin File überschreibt, kann es Speichermäßig günstiger sein, wenn man es vorher direkt löscht. Mit dem " + " Zeichen (CONTROL S), mit dem auch schon initialisiert wurde können die Files auch einzeln gelöscht werden, indem man an den vierstelligen Filenamen einfach das Kreuz anhängt und das File öffnet:

```
OPEN #x,y,z,"Q:FILE+":CLOSE#x
```

oder einfacher mit

```
SAVE "Q:FILE+"
```

In einigen seltenen Fällen kann es vorkommen das die Eindabe in Kurzform (SAVE "Q:XXX") nur noch Fehlermeldungen liefert. Die sist der Fall wenn ein Schreibzugriff auf einen IOCB mit Fehlerstatus stattfindet. Um Abhilfe zu schaffen , geben Sie ein:

```
OPEN #x,y,z,"Q:#":CLOSE#z
```

Weitere Funktionen des QOS sind:

NOTE, POINT, GET, PUT, PRINT, INPUT, ENTER, LIST, RUN

5.0 Basic-Booter Generator

5.1Erklärung

Dieses Programm macht aus BASIC-Files Boot-Disketten. Der " C " & " D "-Handler steht dem Boot-Basic Programm hinterher nicht mehr zur verfügung, was bei den meisten Basic-Programmen jedoch sowieso keine Rolle spielt.

6.0 Tomahawk & Musik

6.1 Erklärung

Tomahawk und Musik sind Demofiles die mit SOUND'N'SAMPLER erstellt wurden. Die 2 nachfolgenden Files auf der Diskette sind die dazugehörigen Datenfiles.

7.0 Let's Frog

7.1 Erklärung

Let's Frog ist ein kleines Spiel das Sie vieleicht in einer anderen Variation schon kennen. Das Ziel debei ist mit dem Frosch die andere Seite zu erreichen indem man die "festen Stellen" zum hüpfen ausnutzt. Um das Spiel zu starten, drücken Sie die " START " Taste. Mit den Tasten " 1 " & " 2 " können Sie die Geschwindigkeit beeinflussen. Die Tastse " 3 " läßt nur eine Fliege erscheinen. Blinkende Frösche können nach Triggerdruck die Fliegen kurzfristig vertreiben.